# Part I

## *General Networking Concepts*

Preview from Notesale.co.uk
Page 4 of 306

All original material copyright © 2014 by Er. Puran S Rawat (superpuran@gmail.com),
unless otherwise noted. All other material copyright © of their respective owners.
This material may be copied and used freely, but may not be altered or sold without the expressed written consent of the owner of the above copyright. Material may be found at http://www.1ccnaprojects.webs.com.

***Network Architectures***

A **host** refers to any device that is connected to a network. A host can also be defined as any device assigned a **network address**.

A host can serve one or more functions:
- A host can *request* data, often referred to as a **client.**
- A host can *provide* data, often referred to as a **server.**
- A host can both request *and* provide data, often referred to as a **peer.**

Because of these varying functions, multiple network **architectures** have been developed, including:
- **Peer-to-Peer**
- **Client/Server**
- **Mainframe/Terminal**

In a basic **peer-to-peer** architecture, all hosts on the network can both *request* and *provide* data and services. For example, two Windows XP workstations configured to share files would be considered a peer-to-peer network.

Peer-to-peer networks are very simple to configure, yet this architecture presents several challenges. Data is difficult to manage and back-up, as it is **spread across multiple devices**. Security is equally problematic, as user accounts and permissions much be configured individually on each host.

Preview from Notesale.co.uk
Page 7 of 306

In a **client/server** architecture, hosts are assigned specific roles. *Clients* request data and services stored on *servers*. An example of a client/server network would be Windows XP workstations accessing files off of a Windows 2003/2008 server.

There are several advantages to the client/server architecture. Data and services are now **centrally located** on one or more servers, consolidating the management and security of that data. As a result, client/server networks can scale far larger than peer-to-peer networks.

One key disadvantage of the client/server architecture is that the server can present a **single point of failure.** This can be mitigated by adding *redundancy* at the server layer.

All original material copyright © 2014 by Er. Puran S Rawat (superpuran@gmail.com), unless otherwise noted. All other material copyright © of their respective owners. This material may be copied and used freely, but may not be altered or sold without the expressed written consent of the owner of the above copyright. Material may be found at http://www.1ccnaprojects.webs.com.

## ***Network Topologies***

A **topology** defines both the *physical* and *logical* structure of a network. Topologies come in a variety of configurations, including:

- Bus
- Star
- Ring
- Full or partial mesh

Ethernet supports two topology types - **bus** and **star**.

## ***Ethernet Bus Topology***

In a **bus topology**, all hosts share a single physical segment (the *bus* or the *backbone*) to communicate:


Preview from Notesale.co.uk
Page 22 of 306

A frame sent by one host is received by *all other* hosts on the bus. However, a host will only *process* a frame if it matches the destination hardware address in the data-link header.

Bus topologies are inexpensive to implement, but are almost entirely deprecated in Ethernet. There are several disadvantages to the bus topology:

- Both ends of the bus must be **terminated,** otherwise a signal will *reflect* back and cause interference, severely degrading performance.
- Adding or removing hosts to the bus can be difficult.
- The bus represents a single point of failure - a break in the bus will affect *all* hosts on the segment. Such faults are often very difficult to troubleshoot.

A bus topology is implemented using either thinnet or thicknet coax cable.

All original material copyright © 2014 by Er. Puran S Rawat (superpuran@gmail.com), unless otherwise noted. All other material copyright © of their respective owners.
This material may be copied and used freely, but may not be altered or sold without the expressed written consent of the owner of the above copyright. Material may be found at http://www.1ccnaprojects.webs.com.

## ***Ethernet (10 Mbps)***

*Ethernet* is now a somewhat generic term, describing the entire family of technologies. However, Ethernet traditionally referred to the original 802.3 standard, which operated at **10 Mbps**. Ethernet supports coax, twisted-pair, and fiber cabling. Ethernet over twisted-pair uses **two** of the four pairs.

Common Ethernet physical standards include:

| ***IEEE Standard*** | ***Physical Standard*** | ***Cable Type*** | ***Maximum Speed*** | ***Maximum Cable Length*** |
|---|---|---|---|---|
| 802.3a | 10base2 | Coaxial (thinnet) | 10 Mbps | 185 meters |
| 802.3 | 10base5 | Coaxial (thicknet) | 10 Mbps | 500 meters |
| 802.3i | 10baseT | Twisted-pair | 10 Mbps | 100 meters |
| 802.3j | 10baseF | Fiber | 10 Mbps | 2000 meters |

Both 10baseT and 10baseF support full-duplex operation, effectively doubling the bandwidth to 20 Mbps. Remember, only a connection **between two hosts** or between **a host and a *switch*** support full-duplex. The maximum distance of an Ethernet segment can be extended through the use of a **repeater**. A hub or a switch can also serve as a repeater.

Preview from Notesale.co.uk
Page 28 of 306

## ***Fast Ethernet (100 Mbps)***

In 1995, the IEEE formalized **802.3u**, a **100 Mbps** revision of Ethernet that became known as **Fast Ethernet.** Fast Ethernet supports both twisted-pair copper and fiber cabling, and supports both half-duplex and full-duplex.

Common Fast Ethernet physical standards include:

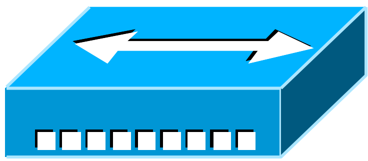
| ***IEEE Standard*** | ***Physical Standard*** | ***Cable Type*** | ***Maximum Speed*** | ***Maximum Cable Length*** |
|---|---|---|---|---|
| 802.3u | 100baseTX | Twisted-pair | 100 Mbps | 100 meters |
| 802.3u | 100baseT4 | Twisted-pair | 100 Mbps | 100 meters |
| 802.3u | 100baseFX | Multimode fiber | 100 Mbps | 400-2000 meters |
| 802.3u | 100baseSX | Multimode fiber | 100 Mbps | 500 meters |

100baseT4 was never widely implemented, and only supported half-duplex operation. 100baseTX is the dominant Fast Ethernet physical standard. 100baseTX uses **two** of the four pairs in a twisted-pair cable, and requires Category 5 cable for reliable performance.

All original material copyright © 2014 by Er. Puran S Rawat (superpuran@gmail.com),
unless otherwise noted. All other material copyright © of their respective owners.
This material may be copied and used freely, but may not be altered or sold without the expressed written consent of the owner of the above copyright. Material may be found at http://www.1ccnaprojects.webs.com.

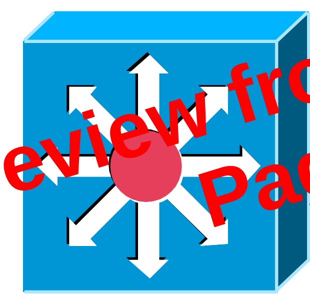
***Icons for Network Devices***

The following icons will be used to represent network devices for all guides on routeralley.com:

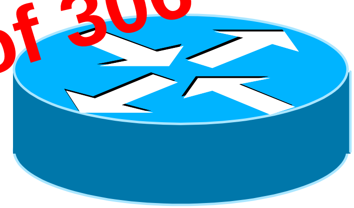


Hub

Switch



Multilayer Switch



Router

Preview from Notesale.co.uk
Page 37 of 306

All original material copyright © 2014 by Er. Puran S Rawat (superpuran@gmail.com), unless otherwise noted. All other material copyright © of their respective owners. This material may be copied and used freely, but may not be altered or sold without the expressed written consent of the owner of the above copyright. Material may be found at http://www.1ccnaprojects.webs.com.

***Layer-2 Switching (continued)***

While hubs were limited to half-duplex communication, switches can operate in **full duplex.** *Each individual port* on a switch belongs to its *own collision domain*. Thus, switches create **more collision domains**, which results in **fewer collisions.**

Like hubs though, switches belong to only *one* **broadcast domain.** A Layer2 switch will forward both broadcasts and multicasts out *every* port but the originating port. Only Layer-3 devices separate broadcast domains.

Because of this, Layer-2 switches are poorly suited for large, scalable networks. The Layer-2 header provides no mechanism to differentiate one *network* from another, only one *host* from another.

This poses *significant* difficulties. If *only* hardware addressing existed, all devices would technically be on the *same* network. Modern internetworks like the Internet could not exist, as it would be impossible to separate *my* network from *your* network.

Imagine if the entire Internet existed purely as a Layer-2 switched environment. Switches, as a rule, will forward a broadcast out *every* port. Even with a conservative estimate of a billion devices on the Internet, the resulting broadcast storms would be devastating. The Internet would simply collapse.

Preview from Notesale.co.uk
Page 40 of 306

Both hubs and switches are susceptible to **switching loops**, which result in destructive broadcast storms. Switches utilize the **Spanning Tree Protocol (STP)** to maintain a loop-free environment. STP is covered in great detail in another guide.

Remember, there are three things that switches do that hubs do not:

- **Hardware address learning**
- **Intelligent forwarding of frames**
- **Loop avoidance**

Hubs are almost entirely deprecated - there is no advantage to using a hub over a switch. At one time, switches were more expensive and introduced more latency (due to processing overhead) than hubs, but this is no longer the case.

All original material copyright © 2014 by Er. Puran S Rawat (superpuran@gmail.com),
unless otherwise noted. All other material copyright © of their respective owners.
This material may be copied and used freely, but may not be altered or sold without the expressed written consent of the owner of the above copyright. Material may be found at http://www.1ccnaprojects.webs.com.

***Extended System IDs***

Normally, a switch's Bridge ID is a 64-bit value that consists of a 16-bit Bridge Priority value, and a 48-bit MAC address.

However, it is possible to include a VLAN ID, called an **extended System ID,** into a Bridge ID. Instead of *adding* bits to the existing Bridge ID, 12 bits of the Bridge Priority value are used for this System ID, which identifies the VLAN this STP process represents.

Because 12 bits have been stolen from the Bridge Priority field, the range of priorities has been reduced. Normally, the Bridge Priority can range from 0 (or *off*) to 65,535, with a default value of 32,768. With extended System ID enabled, the Priority range would be 0 - 61,440, and only in multiples of 4,096.

To enable the extended System ID:

**Switch(config)#** *spanning-tree extend system-id*

Enabling extended System ID accomplishes two things:

- Increases the amount of supported VLANs on the switch from 1005 to 4094.
- Includes the VLAN ID as part of the Bridge ID.

Preview from Notesale.co.uk
Page 54 of 306

Thus, when this command is enabled, the 64-bit Bridge ID will consist of the following:

- 4-bit Priority Value
- 12-bit System ID value (VLAN ID)
- 48-bit MAC address

All original material copyright © 2014 by Er. Puran S Rawat (superpuran@gmail.com), unless otherwise noted. All other material copyright © of their respective owners.
This material may be copied and used freely, but may not be altered or sold without the expressed written consent of the owner of the above copyright. Material may be found at http://www.1ccnaprojects.webs.com.

***STP Timers***

STP utilizes three timers to ensure all switches remain synchronized, and to allow enough time for the Spanning Tree process to ensure a loop-free environment.

- **Hello Timer -** Default is **2 seconds**. Indicates how often BPDU's are sent by switches.
- **Forward Delay -** Default is **15 seconds**. Indicates a *delay* period in both the *listening* and *learning* states of a port, for a total of **30 seconds**. This delay ensures STP has ample time to detect and eliminate loops.
- **Max Age -** Default is **20 seconds**. Indicates how long a switch will keep BPDU information from a neighboring switch before discarding it. In other words, if a switch fails to receive BPDU's from a neighboring switch for the Max Age period, it will remove that switch's information from the STP topology database.

All timer values can be adjusted, and should *only* be adjusted on the Root Bridge. The Root Bridge will **propagate the changed timers** to all other switches participating in STP. Non-Root switches will ignore their locally configured timers.

Preview from Notesale.co.uk
Page 57 of 306

To adjust the three STP timers for VLAN 10:

**Switch(config)#** *spanning-tree vlan 10 hello-time 10*
**Switch(config)#** *spanning-tree vlan 10 forward-time 20*
**Switch(config)#** *spanning-tree vlan 10 max-age 40*

The timers are measured in seconds. The above examples represent the maximum value each timer can be configured to.

Remember that STP is configured on a VLAN by VLAN basis on Catalyst Switches.

All original material copyright © 2014 by Er. Puran S Rawat (superpuran@gmail.com), unless otherwise noted. All other material copyright © of their respective owners.
This material may be copied and used freely, but may not be altered or sold without the expressed written consent of the owner of the above copyright. Material may be found at http://www.1ccnaprojects.webs.com.

# Section 6
# - IPv4 Addressing and Subnetting -

## ***Hardware Addressing***

A **hardware address** is used to uniquely identify a host *within* a local network. Hardware addressing is a function of the **Data-Link** layer of the OSI model (Layer-2).

Ethernet utilizes the 48-bit **MAC address** as its hardware address. The MAC address is often hardcoded on physical network interfaces, though some interfaces support changing the MAC address using special utilities. In virtualization environments, dynamically assigning MAC addresses is very common.

A MAC address is most often represented in **hexadecimal,** using one of two accepted formats:

00:43:AB:F2:32:13
0043.ABF2.3213

Preview from Notesale.co.uk
Page 69 of 306

The first six hexadecimal digits of a MAC address identify the *manufacturer* of the physical network interface. This is referred to as the **OUI (Organizational Unique Identifier).** The last six digits uniquely identify the host itself, and are referred to as the **host ID**.

The MAC address has one shortcoming - it contains no *hierarchy.* MAC addresses provide no mechanism to create **boundaries** between networks. There is no method to distinguish one network from another.

This lack of hierarchy poses *significant* difficulties to network scalability. If *only* Layer-2 hardware addressing existed, all hosts would technically exist on the *same* network. Internetworks like the Internet could not exist, as it would be impossible to separate *my* network from *your* network.

Imagine if the entire Internet existed purely as a single Layer-2 switched network. Switches, as a rule, will forward a broadcast out *every* port. With billions of hosts on the Internet, the resulting broadcast storms would be devastating. The Internet would simply collapse.

The scalability limitations of Layer-2 hardware addresses are mitigated using **logical addresses,** covered in great detail in this guide.

All original material copyright © 2014 by Er. Puran S Rawat (superpuran@gmail.com), unless otherwise noted. All other material copyright © of their respective owners.
This material may be copied and used freely, but may not be altered or sold without the expressed written consent of the owner of the above copyright. Material may be found at http://www.1ccnaprojects.webs.com.

## ***Class A Subnetting Example***

Consider the following subnetted Class A network: 10.0.0.0 255.255.248.0

Now consider the following questions:

- How many new networks were created?
- How many usable hosts are there per network?
- What is the full range of the first three networks?

By default, the *10.0.0.0* network has a subnet mask of *255.0.0.0*. To determine the number of bits stolen:

| | |
|---|---|
| 255.0.0.0: | 11111111.00000000.00000000.00000000 |
| 255.255.248.0: | 11111111.11111111.11111000.00000000 |

Clearly, **13 bits** have been stolen to create the new subnet mask. To calculate the total number of new networks:

$$2^n = 2^{13} = \textbf{8192 new networks created}$$

There are clearly **11 bits** remaining in the host portion of the mask:

$$2^n - 2 = 2^{11} - 2 = 2048 - 2 = \textbf{2046 usable hosts per network}$$

Calculating the ranges is a bit tricky. Using the shortcut method, subtract the third octet (*248*) of the subnet mask (*255.255.248.0*) from *256.*

$$256 - 248 = 8$$

The first network will begin at *0*, again. **However**, the ranges are spread across multiple octets. The ranges of the first three networks look as follows:

| | | | |
|---|---|---|---|
| *Subnet address* | 10.0.0.0 | 10.0.8.0 | 10.0.16.0 |
| *Usable Range* | 10.0.0.1 ↕ 10.0.7.254 | 10.0.8.1 ↕ 10.0.15.254 | 10.0.16.1 ↕ 10.0.23.254 |
| *Broadcast address* | 10.0.7.255 | 10.0.15.255 | 10.0.23.255 |

Preview from Notesale.co.uk
Page 81 of 306

All original material copyright © 2014 by Er. Puran S Rawat (superpuran@gmail.com), unless otherwise noted. All other material copyright © of their respective owners.
This material may be copied and used freely, but may not be altered or sold without the expressed written consent of the owner of the above copyright. Material may be found at http://www.1ccnaprojects.webs.com.

## ***The IPv4 Header***

The IPv4 header is comprised of **12 required fields** and **1 optional field.** The minimum length of the header is **160 bits (20 bytes).**

| *Field* | *Length* | *Description* |
|---|---|---|
| | | |
| Version | 4 bits | *Version of IP (in this case, IPv4)* |
| Internet Header Length | 4 bits | *Specifies the length of the IP header (minimum 160 bits)* |
| DSCP | 8 bits | *Classifies traffic for QoS* |
| Total Length | 16 bits | *Specifies the length of both the header and data payload* |
| Identification | 16 bits | *Uniquely identifies fragments of a packet* |
| Flags | 3 bits | *Flags for fragmentation* |
| Fragment Offset | 13 bits | *Identifies the fragment relative to the start of the packet* |
| Time to Live | 8 bits | *Decremented by each router traversed* |
| Protocol | 8 bits | *Specifies the next upper layer protocol* |
| Header Checksum | 16 bits | *Checksum for error checking* |
| Source Address | 32 bits | *Source IPv4 address* |
| Destination Address | 32 bits | *Destination IPv4 address* |
| Options | Variable | *Optional field for various parameters* |

Preview from Notesale.co.uk
Page 84 of 306

The 4-bit **Version field** is set to a value of *4* for IPv4.

The 4-bit **Internet Header Length field** identifies the length of the IPv4 header, measured in *32-bit words*. The minimum of length of an IPv4 header is 160 bits, or 5 words (32 x 5 = 160).

The 8-bit **Differentiated Service Code Point (DSCP) field** is used to classify traffic for Quality of Service (QoS) purposes. QoS is covered in great detail in other guides. This field was previously referred to as the Type of Service (ToS) field.

The 16-bit **Total Length field** identifies the total packet size, measured in *bytes,* including both the IPv4 header and the data payload. The minimum size of an IPv4 packet is 20 bytes - essentially a header with no payload. The maximum packet size is **65,535 bytes**.

All original material copyright © 2014 by Er. Puran S Rawat (superpuran@gmail.com), unless otherwise noted. All other material copyright © of their respective owners.
This material may be copied and used freely, but may not be altered or sold without the expressed written consent of the owner of the above copyright. Material may be found at http://www.1ccnaprojects.webs.com.

***The IPv4 Header (continued)***

| *Field* | *Length* | *Description* |
|---|---|---|
| | | |
| Version | 4 bits | *Version of IP (in this case, IPv4)* |
| Internet Header Length | 4 bits | *Specifies the length of the IP header (minimum 160 bits)* |
| DSCP | 8 bits | *Classifies traffic for QoS* |
| Total Length | 16 bits | *Specifies the length of both the header and data payload* |
| Identification | 16 bits | *Uniquely identifies fragments of a packet* |
| Flags | 3 bits | *Flags for fragmentation* |
| Fragment Offset | 13 bits | *Identifies the fragment relative to the start of the packet* |
| Time to Live | 8 bits | *Limits the lifetime of a packet* |
| Protocol | 8 bits | *Specifies the next upper layer protocol* |
| Header Checksum | 16 bits | *Checksum for error checking* |
| Source Address | 32 bits | *Source IPv4 address* |
| Destination Address | 32 bits | *Destination IPv4 address* |
| Options | Variable | *Optional field for various parameters* |

Preview from Notesale.co.uk
Page 86 of 306

The 8-bit **Time to Live (TTL) field** limits the lifetime of the packet, preventing it from being endlessly forwarded. When a router forwards a packet, it will decrement the TTL value by one. Once the TTL value reaches zero, the packet is dropped.

The 8-bit **Protocol field** identifies the next upper-layer header, and is covered in the next section.

The 16-bit **Header Checksum field** is used to error-check the IPv4 header. The receiving host will discard the packet if it fails the checksum calculation.

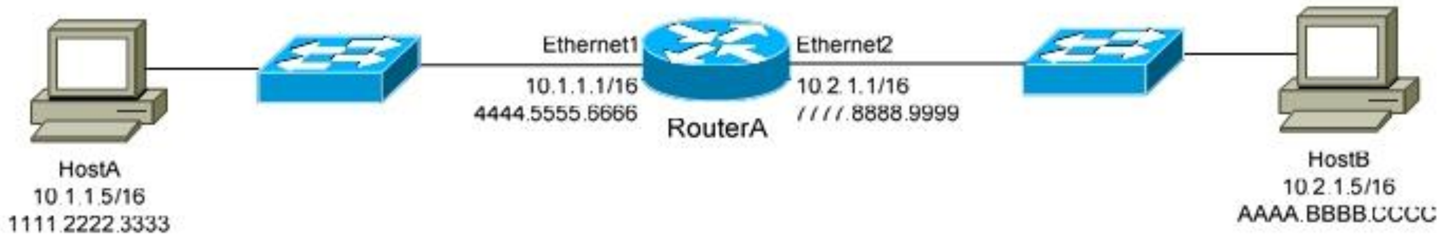
The 32-bit **Source Address field** identifies the *sending* host. The 32-bit **Destination Address field** identifies the *receiving* host. The value of both of these fields can be changed as the packet is forwarded, using NAT.

The variable-length **Options field** provides additional optional IPv4 parameters, outside the scope of this guide.

All original material copyright © 2014 by Er. Puran S Rawat (superpuran@gmail.com), unless otherwise noted. All other material copyright © of their respective owners. This material may be copied and used freely, but may not be altered or sold without the expressed written consent of the owner of the above copyright. Material may be found at http://www.1ccnaprojects.webs.com.

***<u>Resolving Logical Addresses to Hardware Addresses (continued)</u>***

Now consider a slightly modified scenario between HostA and HostB:




- Again, HostA will determine if the destination IP address of 10.2.1.5 is *itself*. In this example, 10.2.1.5 is *not* locally configured on HostA.
- Next, HostA will determine if the 10.2.1.5 address is on the *same network* or *subnet* as itself. In this example, the subnet mask is */16.* Thus, HostA's IP address of 10.1.1.5 and the destination address of 10.2.1.5 are ***not*** on the same network.
- Because HostA and HostB are *not* on the same network, HostA will parse its local routing table for a route to this destination network of 10.2.x.x/16. Hosts are commonly configured with a **default gateway** to reach all other destination networks.
- HostA determines that the 10.1.1.1 address on RouterA is its default gateway. HostA will then broadcast an ARP request, asking for the MAC address of the 10.1.1.1 address.
- RouterA responds to the ARP request with an ARP reply containing its MAC address (4444.5555.6666). HostA can now construct a Layer-2 frame, with a destination of RouterA's MAC address.
- Once RouterA receives the frame, it will parse its own routing table for a route to the destination network of 10.2.x.x/16. It determines that this network is directly attached off of its *Ethernet2* interface. RouterA then broadcasts an ARP request for the 10.2.1.5 address.
- HostB responds to the ARP request with an ARP reply containing its MAC address (AAAA.BBBB.CCCC). RouterA can now construct a Layer-2 frame, with a destination of HostB's MAC address.

Preview from Notesale.co.uk
Page 89 of 306

All original material copyright © 2014 by Er. Puran S Rawat (superpuran@gmail.com), unless otherwise noted. All other material copyright © of their respective owners. This material may be copied and used freely, but may not be altered or sold without the expressed written consent of the owner of the above copyright. Material may be found at http://www.1ccnaprojects.webs.com.

# Section 7
# - TCP and UDP -

## *Transport Layer Protocols*

The **Transport layer (OSI Layer-4)** does *not* actually transport data, despite its name. Instead, this layer is responsible for the *reliable* transfer of data, by ensuring that data arrives at its destination error-free and in order.

The Transport layer is referred to as the **Host-to-Host layer** in the Department of Defense (DoD) reference model.

Transport layer communication falls under two categories:

- **Connection-oriented** - requires that a connection with specific agreed-upon parameters be established before data is sent.
- **Connectionless** - requires no connection before data is sent.

Connection-oriented protocols provide several important services:

- **Connection establishment -** connections are established, maintained, and ultimately terminated between devices.
- **Segmentation and sequencing -** data is *segmented* into smaller pieces for transport. Each segment is assigned a *sequence number*, so that the receiving device can reassemble the data on arrival.
- **Acknowledgments -** receipt of data is confirmed through the use of *acknowledgments*. If a segment is lost, data can be retransmitted to guarantee delivery.
- **Flow control** (or **windowing) -** data transfer rate is negotiated to prevent congestion.

Preview from Notesale.co.uk
Page 92 of 306

The TCP/IP protocol suite incorporates two Transport layer protocols:

- **Transmission Control Protocol (TCP) -** connection-oriented
- **User Datagram Protocol (UDP) -** connectionless

Both TCP and UDP provide a mechanism to differentiate applications running on the same host, through the use of **port numbers.** When a host receives a packet, the port number tells the transport layer which higherlayer application to hand the packet off to.

Both TCP and UDP will be covered in detail in this guide. Please note that the *best* resource on the Internet for TCP/UDP information is the exemplary TCP/IP Guide, found here: http://www.tcpipguide.com/free/index.htm

All original material copyright © 2014 by Er. Puran S Rawat (superpuran@gmail.com), unless otherwise noted. All other material copyright © of their respective owners.
This material may be copied and used freely, but may not be altered or sold without the expressed written consent of the owner of the above copyright. Material may be found at http://www.1ccnaprojects.webs.com.

## ***Port Numbers and Sockets***

Both TCP and UDP provide a mechanism to differentiate applications (or **services**) running on the same host, through the use of **port numbers.** When a host receives a segment, the port number tells the transport layer which higher-layer application to hand the packet off to. This allows multiple network services to operate simultaneously on the same logical address, such as a web *and* an email server.

The range for port numbers is **0 - 65535**, for both TCP and UDP.

The combination of the IP address and port number (identifying both the *host* and *service)* is referred to as a **socket**, and is written out as follows:

192.168.60.125:443

Note the colon separating the IP address (*192.168.60.125*) from the port number (*443*).

The first 1024 ports (**0-1023**) have been reserved for widely-used services, and are recognized as **well-known** ports. Below is a table of several common TCP/UDP ports:

Preview from Notesale.co.uk
Page 93 of 306

| Port Number | Transport Protocol | Application |
|---|---|---|
| 20, 21 | TCP | FTP |
| 22 | TCP | SSH |
| 23 | TCP | Telnet |
| 25 | TCP | SMTP |
| 53 | UDP or TCP | DNS |
| 80 | TCP | HTTP |
| 110 | TCP | POP3 |
| 443 | TCP | SSL |
| 666 | TCP | Doom |

Ports ranging from **1024 - 49151** are referred to as **registered ports**, and are allocated by the IANA upon request. Ports ranging from **49152 - 65535** cannot be registered, and are considered **dynamic.** A client *initiating* a connection will randomly choose a port in this range as its *source* port (for some operating systems, the dynamic range starts at *1024* and higher).

For a complete list of assigned port numbers, refer to the IANA website:

http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xml

All original material copyright © 2014 by Er. Puran S Rawat (superpuran@gmail.com), unless otherwise noted. All other material copyright © of their respective owners.
This material may be copied and used freely, but may not be altered or sold without the expressed written consent of the owner of the above copyright. Material may be found at http://www.1ccnaprojects.webs.com.

## ***TCP Segmentation and Sequencing***

TCP is a **stream-oriented** transport protocol. This allows the application layer to send a continuous stream of unstructured data and rely on TCP to package the data as **segments**, regardless of the amount of data.

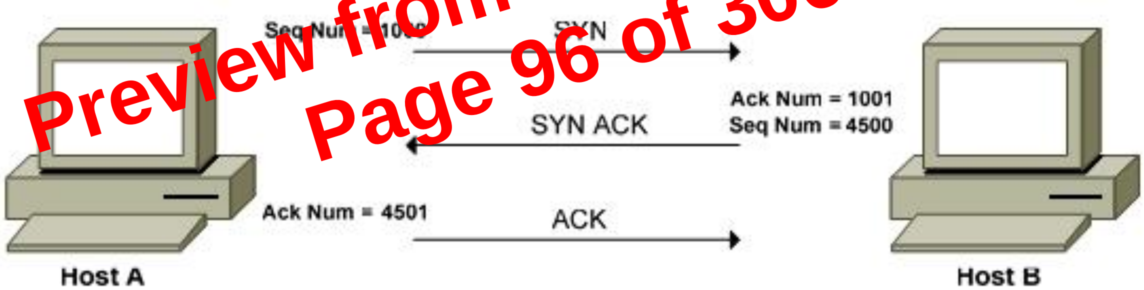
TCP will not only segment data into smaller pieces for transport, but will also assign a **sequence number** to each segment. Note though that this sequence number identifies the data (bytes) *within* the segment rather than the segment *itself.*

Sequencing serves two critical purposes:

- It allows the receiving host to reassemble the data from multiple segments in the correct order, upon arrival.
- It allows receipt of data within a segment to be *acknowledged,* thus providing a mechanism for dropped segments to be detected and resent.

When establishing a connection, a host will choose a 32-bit **initial sequence number (ISN).** The ISN is chosen from a randomizing timer, to prevent accidental overlap or predictability.

Preview from Notesale.co.uk
Page 96 of 306




The receiving host responds to this sequence number with an **acknowledgment number**, set to the **sequence number + 1**. In the above example, HostB's acknowledgment number would thus be *1001*.

HostB includes an initial sequence number with *its* SYN message as well - *4500* in the above example. HostA would respond to this sequence number with an acknowledgement number of *4501.*

The TCP header contains both a 32-bit Sequence Number and 32-bit Acknowledgement Number field.

All original material copyright © 2014 by Er. Puran S Rawat (superpuran@gmail.com), unless otherwise noted. All other material copyright © of their respective owners.
This material may be copied and used freely, but may not be altered or sold without the expressed written consent of the owner of the above copyright. Material may be found at http://www.1ccnaprojects.webs.com.

***The TCP Header (continued)***

| ***Field*** | ***Length*** | ***Description*** |
|---|---|---|
| | | |
| Source Port | 16 bits | *Source TCP Port* |
| Destination Port | 16 bits | *Destination TCP Port* |
| Sequence Number | 32 bits | *Sequence Number* |
| Ack Number | 32 bits | *Acknowledgement Number* |
| Data Offset | 4 bits | *Indicates where the data begins in a TCP segment* |
| Reserved | 6 bits | *Always set to 0* |
| Control Bits | 6 bits | *URG, ACK, PSH, RST, SYN, and FIN flags* |
| Window | 16 bits | *Used for Flow Control* |
| Checksum | 16 bits | *Used for Error-Checking* |
| Urgent Pointer | 16 bits | *Identifies last byte of Urgent traffic* |
| Options | Variable | |
| Padding | Variable | *To ensure the TCP header ends at a 32-bit boundary* |

The 6-bit **Control Bits** field contains six 1-bit flags in the following order:

- **URG (Urgent) -** prioritizes specified traffic.
- **ACK (Acknowledgment) -** acknowledges a SYN or receipt of data.
- **PSH (Push) -** forces an immediate send even if window is not full.
- **RST (Reset) -** forcefully terminates an improper connection.
- **SYN (Synchronize) -** initiates a connection.
- **FIN (Finish) -** gracefully terminates a connection when there is further data to send.

Preview from Notesale.co.uk Page 103 of 306

The 16-bit **Window field** identifies the number of data octets that the receiver is able to accept.

The 16-bit **Checksum field** is used for error-checking, and is computed using both the TCP segment and select fields from the IP header. The receiving host will discard the segment if it fails the checksum calculation.

The 16-bit **Urgent Pointer field** is used to identify the *last* byte of prioritized traffic in a segment, when the URG flag is set.

The variable-length **Options field** provides additional optional TCP parameters, outside the scope of this guide.

The variable-length **Padding field** ensures the TCP header ends on a 32-bit boundary, and is always set to zeroes.

All original material copyright © 2014 by Er. Puran S Rawat (superpuran@gmail.com), unless otherwise noted. All other material copyright © of their respective owners. This material may be copied and used freely, but may not be altered or sold without the expressed written consent of the owner of the above copyright. Material may be found at http://www.1ccnaprojects.webs.com.

## ***User Datagram Protocol (UDP)***

The **User Datagram Protocol (UDP)** is a **connectionless** transport protocol, and is defined in RFC 768.

UDP, above all, is *simple.* It provides no three-way handshake, no flowcontrol, no sequencing, and no acknowledgment of data receipt. UDP essentially forwards the segment and takes no further interest.

Thus, UDP is **inherently unreliable,** especially compared to a connectionoriented protocol like TCP. However, UDP **experiences less latency** than TCP, due to the reduced overhead. This makes UDP ideal for applications that require speed over reliability. For example, DNS primarily uses UDP as its transport protocol, though it supports TCP as well.

Like TCP, UDP does provide basic error-checking using a checksum, and uses port numbers to differentiate applications running on the same host.

The UDP header has only 4 **fields**:

| ***Field*** | ***Length*** | ***Description*** |
|---|---|---|
| | | |
| Source Port | 16 bits | *Source UDP Port* |
| Destination Port | 16 bits | *Destination UDP Port* |
| Length | 16 bits | *Length of the header and the data* |
| Checksum | 16 bits | *Used for Error-Checking* |

Preview from Notesale.co.uk
Page 104 of 306

The following provides a quick comparison of TCP and UDP:

| ***TCP*** | ***UDP*** |
|---|---|
| Connection-oriented | Connectionless |
| Guarantees delivery | Does *not* guarantee delivery |
| Sends acknowledgments | Does *not* send acknowledgments |
| Reliable, but slower than UDP | Unreliable, but faster than TCP |
| Segments and sequences data | Does *not* provide sequencing |
| Resends dropped segments | Does *not* resend dropped segments |
| Provides flow control | Does *not* provide flow control |
| Performs CRC on data | Also performs CRC on data |
| Uses port numbers | Also uses port numbers |

All original material copyright © 2014 by Er. Puran S Rawat (superpuran@gmail.com), unless otherwise noted. All other material copyright © of their respective owners.
This material may be copied and used freely, but may not be altered or sold without the expressed written consent of the owner of the above copyright. Material may be found at http://www.1ccnaprojects.webs.com.

## ***The IPv6 Address***

The IPv6 address is **128 bits**, as opposed to the 32-bit IPv4 address. Also unlike IPv4, the IPv6 address is represented in hexadecimal notation, separate by colons.

An example of an IPv6 address would be:

1254:1532:26B1:CC14:0123:1111:2222:3333

Each "grouping" (from here on called **fields**) of hexadecimal digits is 16 bits, with a total of eight fields. The hexadecimal values of an IPv6 address are **not case-sensitive.**

We can drop any leading zeros in each field of an IPv6 address. For example, consider the following address:

1423:0021:0C13:CC1E:3142:0001:2222:3333

We can condense that address to: 1423:21:C13:CC1E:3142:1:2222:3333

*Only* leading zeros can be condensed. If we have an entire field comprised of zeros, we can further compact the following address:

F12F:0000:0000:CC1E:2412:1111:2222:3333

The condensed address would be: F12F::CC1E:2412:1111:2222:3333

Notice the double colons (::). We can only condense one set of contiguous zero fields. Thus, if we had the following address:

F12F:0000:0000:CC1E:2412:0000:0000:3333

We could not condense that to: F12F::CC1E:2412::3333

The address would now be ambiguous, as we wouldn't know how many "0" fields were compacted in each spot. Remember that we can only use one set of double colons in an IPv6 address!

Preview from Notesale.co.uk
Page 106 of 306

All original material copyright © 2014 by Er. Puran S Rawat (superpuran@gmail.com), unless otherwise noted. All other material copyright © of their respective owners.
This material may be copied and used freely, but may not be altered or sold without the expressed written consent of the owner of the above copyright. Material may be found at http://www.1ccnaprojects.webs.com.

## ***The IPv6 Address Hierarchy***

IPv4 separated its address space into specific **classes.** The class of an IPv4 address was identified by the high-order bits of the first octet:

- **Class A** - (00000001 - 01111111, or 1 - 127)
- **Class B -** (10000000 - 10111111, or 128 - 191)
- **Class C -** (11000000 - 11011111, or 192 - 223)
- **Class D -** (11100000 - 11101111, or 224 - 239)

IPv6's addressing structure is far more scalable. Less than 20% of the IPv6 address space has been designated for use, currently. The potential for growth is enormous.

The address space that *has* been allocated is organized into several types, determined by the high-order bits of the first field:

- **Special Addresses -** addresses begin **00*xx*:**

- **Link Local -** addresses begin **FE8*x*:**
- **Site Local -** addresses begin **FEC*x*:**
- **Aggregate Global -** addresses begin **2*xxx*:** or **3*xxx*:**
- **Multicasts -** addresses begin **FF*xx*:**
- **Anycasts**

*(Note: an "x" indicates the value can be any hexadecimal number)*

Preview from Notesale.co.uk
Page 108 of 306

There are **no broadcast addresses** in IPv6. Thus, any IPv6 address that is not a *multicast* is a *unicast* address.

**Anycast addresses** identify a group of interfaces on multiple hosts. Thus, multiple hosts are configured with an *identical* address. Packets sent to an anycast address are sent to the *nearest* (i.e., least amount of hops) host. Anycasts are indistinguishable from any other IPv6 unicast address.

Practical applications of anycast addressing are a bit murky. One possible application would be a server farm providing an identical service or function, in which case anycast addressing would allow clients to connect to the nearest server.

All original material copyright © 2014 by Er. Puran S Rawat (superpuran@gmail.com), unless otherwise noted. All other material copyright © of their respective owners. This material may be copied and used freely, but may not be altered or sold without the expressed written consent of the owner of the above copyright. Material may be found at http://www.1ccnaprojects.webs.com.

## ***Aggregate Global IPv6 Addresses***

**Aggregate Global IPv6 addresses** are the equivalent of "public" IPv4 addresses. Aggregate global addresses can be routed publicly on the Internet. Any device or site that wishes to traverse the Internet must be uniquely identified with an aggregate global address.

Currently, the first field of an **aggregate global** IPv6 address will always begin **2*xxx* (001).** Aggregate global addresses are **unicasts,** and represent 1/8th of the available IPv6 address space.

**2000::2731:E2FF:FE96:C283/64**

Aggregate global addresses adhere to a very strict hierarchy:

- The first 3 bits are the fixed FP.
- The next 13 bits are the **top-level aggregation identifier (TLA ID).**
- The next 8 bits are **reserved** for future use.
- The next 24 bits are the **next-level aggregation identifier (NLA ID).**
- The next 16 bits are the **site-level aggregation identifier (SLA ID).**
- The final 64 bits are used as the **interface ID**.

By have multiple **levels**, a consistent, organized, and scalable hierarchy is maintained. High-level registries are assigned ranges of TLA IDs. These can then be subdivided in the NLA ID field, and passed on to lower-tiered ISPs.

Preview from Notesale.co.uk
Page 111 of 306

Such ISPs allocate these prefixes to their customers, which can further subdivide the prefix using the SLA ID field, to create whatever local hierarchy they wish. The 16-bit SLA field provides up to 65535 networks for an organization.

Note: Do not confuse the SLA ID field of a global address field, with a site-local address. Site-local addresses cannot be routed publicly, where as SLA ID's are just a subset of the publicly routable aggregate global address.

All original material copyright © 2014 by Er. Puran S Rawat (superpuran@gmail.com), unless otherwise noted. All other material copyright © of their respective owners.
This material may be copied and used freely, but may not be altered or sold without the expressed written consent of the owner of the above copyright. Material may be found at http://www.1ccnaprojects.webs.com.

# Section 9
# - Introduction to 802.11 Wireless -

***802.11 Overview***

In the mid 1990's, the IEEE LAN/MAN committee began developing a series of **Wireless Local Area Network (WLAN)** standards. Collectively, these wireless standards are identified as the **802.11 standard.**

**Note:** The 802.11 standard is occasionally referred to as **Wi-Fi**, though the term 'Wi-Fi' has been applied to other wireless standards as well.

Various **amendments** have been made to the 802.11 standard. These are identified by the letter appended to the standard, such as 802.11*a* or 802.11*g*. The 802.11 amendments will be covered in greater detail later in this guide.

Wireless devices communicate across a specific range of **RF frequencies** known as a **channel**, using an antenna off of a **radio card**. 802.11 antennas come in several forms:

- Omnidirectional
- Semi-directional
- Highly-directional

Preview from Notesale.co.uk
Page 119 of 306

A group of communicating 802.11 wireless devices is known as a **service set.** A wireless client can connect point-to-point with another wireless client - this is referred to as an **ad-hoc** connection, or an **Independent Basic Service Set (IBSS).**

More commonly, wireless client are centrally connected via a **wireless access point (WAP).** This is referred to as an **infrastructure** connection, or a **Basic Service Set (BSS)**. Wireless clients must **associate** with a WAP before data can be forwarded. WAPs often serve as a *gateway* between the wired and wireless networks.

In environments where a single WAP does not provide sufficient coverage, multiple WAPs can be *linked* as part of an **Extended Service Set (ESS)**.

All original material copyright © 2014 by Er. Puran S Rawat (superpuran@gmail.com), unless otherwise noted. All other material copyright © of their respective owners.
This material may be copied and used freely, but may not be altered or sold without the expressed written consent of the owner of the above copyright. Material may be found at http://www.1ccnaprojects.webs.com.

## ***802.11b***

The **802.11b** amendment was also released in 1999, and utilizes **complementary code keying (CCK)** for modulation. 802.11b operates in the **2.4-GHz** frequency band, and has a *maximum* throughput of **11 Mbps.** Specifically, 802.11b supports data rates of 1, 2, 5.5, and 11 Mbps.

Because 802.11b operates in the unregulated 2.4-GHz band, it is susceptible to interference from other household RF devices.

In the U.S., 802.11b supports a total of **3** non-overlapping channels, specifically channels **1, 6,** and **11**.

(Reference: http://en.wikipedia.org/wiki/IEEE_802.11b-1999)

Preview from Notesale.co.uk
Page 128 of 306

## ***802.11g***

The **802.11g** amendment was released in 2003, and utilizes **orthogonal frequency-division multiplexing (OFDM)** for modulation. 802.11g operates in the **2.4-GHz** frequency band, and has a *maximum* throughput of **54 Mbps**. Specifically, 802.11g supports data rates of 6, 9, 12, 18, 24, 36, 48, and 54 Mbps.

As with 802.11b, 802.11g operates in the unregulated 2.4-GHz band, and is susceptible to interference from other household RF devices.

In the U.S., 802.11g supports a total of **3** non-overlapping channels, specifically channels **1, 6,** and **11**.

802.11g is **backward-compatible** with 802.11b, as they both operate in the 2.4-GHz band. However, if an 802.11b device is present in an 802.11g environment, 802.11g will revert to CCK modulation, and will only support throughputs of 1, 2, 5.5, and 11 Mbps.

Neither 802.11b nor 802.11g are backward-compatible with 802.11a.

(Reference: http://en.wikipedia.org/wiki/IEEE_802.11g-2003)

All original material copyright © 2014 by Er. Puran S Rawat (superpuran@gmail.com), unless otherwise noted. All other material copyright © of their respective owners.
This material may be copied and used freely, but may not be altered or sold without the expressed written consent of the owner of the above copyright. Material may be found at http://www.1ccnaprojects.webs.com.

## ***802.1X and Extensible Authentication Protocol (EAP)***

The **802.1X standard** was developed by the IEEE to authenticate devices on a Layer-2 port basis. It was originally developed for Ethernet (802.3) bridges and switches, but was expanded to support the authentication of 802.11 wireless devices as well.

802.1X defines three *roles* in the authentication process:

- **Supplicant** - the device being authenticated. In an 802.11 environment, the supplicant would be the wireless client software.
- **Authenticator** - the device that is *requiring* the authentication. In an 802.11 environment, this is often the WAP.
- **Authentication Server** - the device that stores the user database, for validating authentication credentials. This is often an external RADIUS server, though some WAPs support a local user database.

802.1X provides the *encapsulation* of **Extensible Authentication Protocol (EAP)** traffic, which serves as the *framework* for authenticating clients. EAP is not an authentication mechanism in itself. Instead, EAP transports the authentication data between supplicants, authenticators, and authentication servers (all three of which must support 802.1X/EAP).

Preview from Notesale.co.uk
Page 133 of 306

As a general framework, EAP supports a large number of *methods* for authentication, including (but not limited to):

- **Lightweight EAP (LEAP)**
- **EAP - Flexible Authentication via Secure Tunneling (EAP-FAST)**
- **EAP - Transport Layer Security (EAP-TLS)**
- **Protected EAP (PEAP)**

With any form of EAP, wireless clients *must* authenticate with a RADIUS server before any data traffic will be forwarded. Only EAP traffic is allowed between the client and WAP before authentication occurs.

Authenticating clients using 802.1X/EAP offers several advantages over Static-WEP and WPA-PSK, including:

- Centralized management of credentials
- Support for multiple encryption types
- Dynamic encryption keys

(Reference: http://en.wikipedia.org/wiki/IEEE_802.1X; http://en.wikipedia.org/wiki/Extensible_Authentication_Protocol; http://www.ieee802.org/1/files/public/docs2000/P8021XOverview.PDF)

All original material copyright © 2014 by Er. Puran S Rawat (superpuran@gmail.com), unless otherwise noted. All other material copyright © of their respective owners.
This material may be copied and used freely, but may not be altered or sold without the expressed written consent of the owner of the above copyright. Material may be found at http://www.1ccnaprojects.webs.com.

## ***Using Lines to Configure the IOS***

As mentioned previously, three methods (or *lines*) exist to configure Cisco IOS devices:

- Console ports
- Auxiliary ports
- VTY (telnet) ports

Nearly every modern Cisco router or switch includes a **console port,** sometimes labeled on the device simply as *con*. The console port is generally a RJ-45 connector, and requires a **rollover** cable to connect to. The opposite side of the rollover cable connects to a PC's serial port using a serial **terminal adapter**.

From the PC, software such as HyperTerminal is required to make a connection from the local serial port to the router console port. The following settings are necessary for a successful connection:

- Bits per second - *9600 baud*
- Data bits - *8*
- Parity - *None*
- Stop bits - *1*
- Flow Control - *Hardware*

Preview from Notesale.co.uk
Page 141 of 306

Some Cisco devices include an **auxiliary port,** in addition to the console port. The auxiliary port can function similarly to a console port, and can be accessed using a rollover cable. Additionally, auxiliary ports support modem commands, thus providing dial-in access to Cisco devices.

**Telnet**, and now **SSH**, are the most common methods of remote access to routers and switches. The standard edition of the IOS supports up to **5 simultaneous** VTY connections. Enterprise editions of the IOS support up to **255 VTY** connections.

There are two requirements before a router/switch will accept a VTY connection:

- An **IP address** must be configured on an interface
- At least one VTY port must be configured with a **password**

All original material copyright © 2014 by Er. Puran S Rawat (superpuran@gmail.com), unless otherwise noted. All other material copyright © of their respective owners.
This material may be copied and used freely, but may not be altered or sold without the expressed written consent of the owner of the above copyright. Material may be found at http://www.1ccnaprojects.webs.com.

## ***Status of Router Interfaces (continued)***

Traffic can only be routed across an interface if its status is as follows:

***Serial 0 is up, line protocol is up***

The first part of this status (*Serial0 is up)* refers to the **physical layer** status of the interface. The second part (*line protocol is up)* refers to the **data-link layer** status of the interface. A status of *up/up* indicates that the physical interface is active, and both sending and receiving keepalives.

An interface that is physically down will display the following status:

***Serial 0 is down, line protocol is down***

The mostly likely cause of the above status is a defective (or unplugged) cable or interface.

There are several potential causes of the following status:

***Serial 0 is up, line protocol is down***

Preview from Notesale.co.uk Page 154 of 306

Recall that *line protocol* refers to data-link layer functions. Potential causes of the above status could include:

- Absence of keepalives being sent or received
- Clock rate not set on the DCE side of a serial connection
- Different encapsulation types set on either side of the link

An interface that has been administratively shutdown will display the following status:

***Serial 0 is administratively down, line protocol is down***

All original material copyright © 2014 by Er. Puran S Rawat (superpuran@gmail.com), unless otherwise noted. All other material copyright © of their respective owners. This material may be copied and used freely, but may not be altered or sold without the expressed written consent of the owner of the above copyright. Material may be found at http://www.1ccnaprojects.webs.com.

# Section 14
# - Static vs. Dynamic Routing -

***Static vs. Dynamic Routing***

There are two basic methods of building a routing table: **Statically** or **Dynamically**.

A **static** routing table is created, maintained, and updated by a network administrator, *manually*. A static route to *every* network must be configured on *every* router for full connectivity. This provides a granular level of control over routing, but quickly becomes impractical on large networks.

Routers will *not* share static routes with each other, thus reducing CPU/RAM overhead and saving bandwidth. However, static routing is *not fault-tolerant*, as any change to the routing infrastructure (such as a link going down, or a new network added) requires manual intervention. Routers operating in a purely static environment cannot seamlessly choose a better route if a link becomes unavailable.

Static routes have an Administrative Distance (AD) of 1, and thus are always preferred over dynamic routes, unless the default AD is changed. A static route with an adjusted AD is called a **floating static route**, and is covered in greater detail in another guide.

Preview from Notesale.co.uk
Page 167 of 306

A **dynamic** routing table is created, maintained, and updated by a *routing protocol* running on the router. Examples of routing protocols include **RIP** (Routing Information Protocol), **EIGRP** (Enhanced Interior Gateway Routing Protocol), and **OSPF** (Open Shortest Path First). Specific dynamic routing protocols are covered in great detail in other guides.

Routers *do* share dynamic routing information with each other, which increases CPU, RAM, and bandwidth usage. However, routing protocols are capable of dynamically choosing a different (or better) path when there is a change to the routing infrastructure.

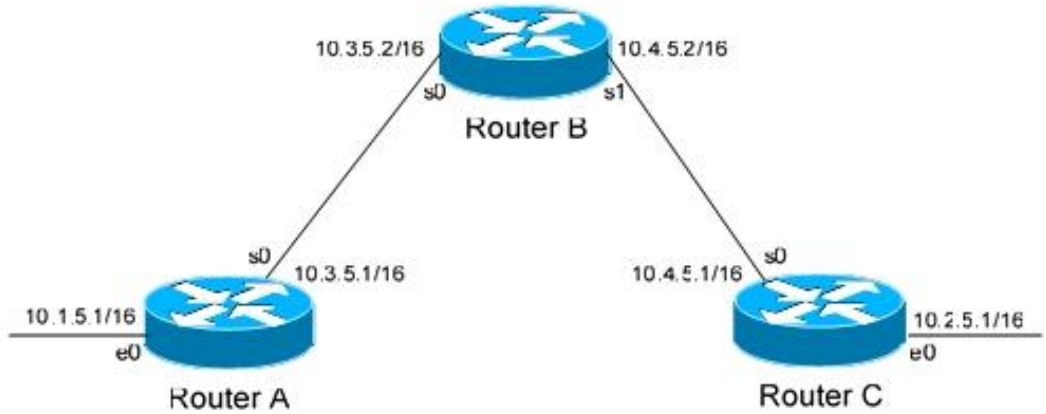
Do not confuse *routing* protocols with *routed* protocols:

- A ***routed*** protocol is a Layer 3 protocol that applies logical addresses to devices and routes data between networks (such as IP)
- A ***routing*** protocol dynamically builds the network, topology, and next hop information in routing tables (such as RIP, EIGRP, etc.)

All original material copyright © 2014 by Er. Puran S Rawat (superpuran@gmail.com), unless otherwise noted. All other material copyright © of their respective owners.
This material may be copied and used freely, but may not be altered or sold without the expressed written consent of the owner of the above copyright. Material may be found at http://www.1ccnaprojects.webs.com.

## ***Limitations of Classful Routing Example***

The following section will illustrate the limitations of classful routing, using RIPv1 as an example. Consider the following diagram:




This particular scenario will work when using RIPv1, despite the fact that we've subnetted the major 10.0.0.0 network. Notice that the subnets are contiguous (that is, they belong to the same major network), and use the same subnet mask.

When Router A sends a RIPv1 update to Router B via Serial0, it will not include the subnet mask for the 10.1.0.0 network. However, because the 10.3.0.0 network is in the same major network as the 10.1.0.0 network, it will **not summarize** the address. The route entry in the update will simply state "10.1.0.0".

Preview from Notesale.co.uk Page 173 of 306

Router B will accept this routing update, and realize that the interface receiving the update (Serial0) belongs to the same major network as the route entry of 10.1.0.0. It will then apply the subnet mask of its Serial0 interface to this route entry.

Router C will similarly send an entry for the 10.2.0.0 network to Router B. Router B's routing table will thus look like:

**RouterB#** *show ip route*

```
Gateway of last resort is not set

       10.0.0.0/16 is subnetted, 4 subnets
C      10.3.0.0 is directly connected, Serial0
C      10.4.0.0 is directly connected, Serial1
R      10.1.0.0 [120/1] via 10.3.5.1, 00:00:00, Serial0
R      10.2.0.0 [120/1] via 10.4.5.1, 00:00:00, Serial1
```

All original material copyright © 2014 by Er. Puran S Rawat (superpuran@gmail.com), unless otherwise noted. All other material copyright © of their respective owners. This material may be copied and used freely, but may not be altered or sold without the expressed written consent of the owner of the above copyright. Material may be found at http://www.1ccnaprojects.webs.com.

## ***RIP Timers Configuration and Example***


Consider the above example. Router A receives a RIP update from Router B that includes network 172.18.0.0. Router A adds this network to its routing table:

**RouterA#** *show ip route*

```
Gateway of last resort is not set

C      172.16.0.0 is directly connected, Ethernet0
C      172.17.0.0 is directly connected, Serial0
R      172.18.0.0 [120/1] via 172.17.1.2, 00:00:00, Serial0
```

Immediately, Router A sets an **invalid** timer of 180 seconds and **flush** timer of 240 seconds to this route, which run concurrently. If no update for this route is heard for 180 seconds, several things will occur:

- The route is marked as invalid in the routing table.
- The route enters a **hold-down** state (triggering the hold-down timer).
- The route is advertised to all other routers as unreachable.

The hold-down timer runs for 180 seconds *after* the route is marked as invalid. The router will not accept any new updates for this route until this hold-down period expires.

If no update is heard *at all*, the route will be removed from the routing table once the flush timer expires, which is 60 seconds after the route is marked as invalid. Remember that the invalid and flush timers run concurrently.

To configure the RIP timers:

**Router(config)#** *router rip*
**Router(config-router)#** *timers basic 20 120 120 160*

The *timers basic* command allows us to change the update (*20*), invalid (*120*), hold-down (*120)*, and flush *(240)* timers. To return the timers back to their defaults:

**Router(config-router)#** *no timers basic*

Preview from Notesale.co.uk
Page 186 of 306

All original material copyright © 2014 by Er. Puran S Rawat (superpuran@gmail.com), unless otherwise noted. All other material copyright © of their respective owners.
This material may be copied and used freely, but may not be altered or sold without the expressed written consent of the owner of the above copyright. Material may be found at http://www.1ccnaprojects.webs.com.

## ***Configuring IGRP***


Routing protocol configuration occurs in Global Configuration mode. On Router A, to configure IGRP, we would type:

**Router(config)#** *router igrp 10*
**Router(config-router)#** *network 172.16.0.0*
**Router(config-router)#** *network 172.17.0.0*

The first command, *router igrp 10*, enables the IGRP process. The "*10"* indicates the Autonomous System number that we are using. Only other IGRP routers in Autonomous System *10* will share updates with this router.

The *network* statements tell IGRP which networks you wish to advertise to other RIP routers. We simply list the networks that are directly connected to our router. Notice that we specify the networks at their classful boundaries, and we do not specify a subnet mask.

To configure Router B:

Preview from Notesale.co.uk
Page 194 of 306

**Router(config)#** *router igrp 10*
**Router(config-router)#** *network 172.17.0.0*
**Router(config-router)#** *network 172.18.0.0*

The routing table on Router A will look like:

**RouterA#** *show ip route*

```
Gateway of last resort is not set

C     172.16.0.0 is directly connected, Ethernet0
C     172.17.0.0 is directly connected, Serial0
I     172.18.0.0 [120/1] via 172.17.1.2, 00:00:00, Serial0
```

The routing table on Router B will look like:

**RouterB#** *show ip route*

```
Gateway of last resort is not set

C     172.17.0.0 is directly connected, Serial0
C     172.18.0.0 is directly connected, Ethernet0
I     172.16.0.0 [120/1] via 172.17.1.1, 00:00:00, Serial0
```

All original material copyright © 2014 by Er. Puran S Rawat (superpuran@gmail.com), unless otherwise noted. All other material copyright © of their respective owners.
This material may be copied and used freely, but may not be altered or sold without the expressed written consent of the owner of the above copyright. Material may be found at http://www.1ccnaprojects.webs.com.

## ***EIGRP Packet Types***

EIGRP employs five packet types:

- **Hello packets -** *multicast*
- **Update packets -** *unicast* or *multicast*
- **Query packets** - *multicast*
- **Reply packets** - *unicast*
- **Acknowledgement packets -** *unicast*

**Hello packets** are used to form neighbor relationships, and were explained in detail previously. Hello packets are always multicast to address 224.0.0.10.

**Update packets** are sent between neighbors to build the topology and routing tables. Updates sent to *new* neighbors are sent as unicasts. However, if a route's metric is changed, the update is sent out as a multicast to address 224.0.0.10.

**Query packets** are sent by a router when a Successor route fails, and there are no Feasible Successors in the topology table. The router places the route in an **Active state**, and queries its neighbors for an alternative route. Query packets are sent as a multicast to address 224.0.0.10.

**Reply packets** are sent in response to Query packets, assuming the responding router has an alternative route (feasible successor). Reply packets are sent as a unicast to the querying router.

Preview from Notesale.co.uk
Page 207 of 306

Recall that EIGRP utilizes the **Reliable Transport Protocol** (**RTP**) to ensure reliable delivery of most EIGRP packets. Delivery is guaranteed by having packets *acknowledged* using…..**Acknowledgment packets!**

Acknowledgment packets (also known as **ACK's**) are simply Hello packets with no data, other than an acknowledgment number. ACK's are always sent as unicasts. The following packet types employ RTP to ensure reliable delivery via ACK's:

- Update Packets
- Query Packets
- Reply Packets

Hello and Acknowledgments (ha!) packets do not utilize RTP, and thus do not require acknowledgement.

All original material copyright © 2014 by Er. Puran S Rawat (superpuran@gmail.com), unless otherwise noted. All other material copyright © of their respective owners.
This material may be copied and used freely, but may not be altered or sold without the expressed written consent of the owner of the above copyright. Material may be found at http://www.1ccnaprojects.webs.com.

## ***OSPF Neighbors***

OSPF forms neighbor relationships, called **adjacencies,** with other routers in the same **Area** by exchanging **Hello** packets to multicast address **224.0.0.5**. Only after an adjacency is formed can routers share routing information.

Each OSPF router is identified by a unique **Router ID.** The Router ID can be determined in one of three ways:

- The Router ID can be **manually** specified.
- If not manually specified, the highest IP address configured on any **Loopback interface** on the router will become the Router ID.
- If no loopback interface exists, the highest IP address configured on any **Physical interface** will become the Router ID.

By default, Hello packets are sent out OSPF-enabled interfaces every **10 seconds** for broadcast and point-to-point interfaces, and **30 seconds** for nonbroadcast and point-to-multipoint interfaces.

OSPF also has a **Dead Interval**, which indicates how long a router will wait without hearing any hellos before announcing a neighbor as "down." Default for the Dead Interval is **40 seconds** for broadcast and point-to-point interfaces, and **120 seconds** for non-broadcast and point-to-multipoint interfaces. Notice that, by default, the dead interval timer is four times the Hello interval.

Preview from Notesale.co.uk
Page 219 of 306

These timers can be adjusted on a *per interface* basis:

**Router(config-if)#** *ip ospf hello-interval 15*
**Router(config-if)#** *ip ospf dead-interval 60*

All original material copyright © 2014 by Er. Puran S Rawat (superpuran@gmail.com), unless otherwise noted. All other material copyright © of their respective owners.
This material may be copied and used freely, but may not be altered or sold without the expressed written consent of the owner of the above copyright. Material may be found at http://www.1ccnaprojects.webs.com.

## ***OSPF Neighbor States***

Neighbor adjacencies will progress through several **states**, including:

**Down** - indicates that no Hellos have been heard from the neighboring router.

**Init -** indicates a Hello packet has been heard from the neighbor, but twoway communication has not yet been initialized.

**2-Way** - indicates that bidirectional communication has been established. Recall that Hello packets contain a *neighbor* field. Thus, communication is considered 2-Way once a router sees its own Router ID in its neighbor's Hello Packet. **Designated** and **Backup Designated Routers** are elected at this stage.

**ExStart** - indicates that the routers are preparing to share link state information. Master/slave relationships are formed between routers to determine who will begin the exchange.

Preview from Notesale.co.uk
Page 222 of 306

**Exchange -** indicates that the routers are exchanging **Database Descriptors (DBDs).** DBDs contain a description of the router's Topology Database. A router will examine a neighbor's DBD to determine if it has information to share.

**Loading -** indicates the routers are finally exchanging **Link State Advertisements,** containing information about all links connected to each router. Essentially, routers are sharing their topology tables with each other.

**Full -** indicates that the routers are fully synchronized. The topology table of all routers in the area should now be identical. Depending on the "role" of the neighbor, the state may appear as:

- **Full/DR** - indicating that the neighbor is a Designated Router (DR)
- **Full/BDR** - indicating that the neighbor is a Backup Designated Router (BDR)
- **Full/DROther -** indicating that the neighbor is neither the DR or BDR

On a multi-access network, OSPF routers will *only* form Full adjacencies with DRs and BDRs. Non-DRs and non-BDRs will still form adjacencies, but will remain in a **2-Way State**. This is normal OSPF behavior.

All original material copyright © 2014 by Er. Puran S Rawat (superpuran@gmail.com), unless otherwise noted. All other material copyright © of their respective owners.
This material may be copied and used freely, but may not be altered or sold without the expressed written consent of the owner of the above copyright. Material may be found at http://www.1ccnaprojects.webs.com.

## ***OSPF Network Types***

OSPF's functionality is different across several different network topology types. OSPF's interaction with Frame Relay will be explained in another section

**Broadcast Multi-Access -** indicates a topology where broadcast occurs.

- Examples include Ethernet, Token Ring, and ATM.
- OSPF *will* elect DRs and BDRs.
- Traffic to DRs and BDRs is multicast to 224.0.0.6. Traffic from DRs and BDRs to other routers is multicast to 224.0.0.5.
- Neighbors *do not* need to be manually specified.

**Point-to-Point -** indicates a topology where two routers are directly connected.

- An example would be a point-to-point T1.
- OSPF *will not* elect DRs and BDRs.
- All OSPF traffic is multicast to 224.0.0.5.
- Neighbors *do not* need to be manually specified.

Preview from Notesale.co.uk
Page 223 of 306

**Point-to-Multipoint** - indicates a topology where one interface can connect to multiple destinations. Each connection between a source and destination is treated as a point-to-point link.

- An example would be Point-to-Multipoint Frame Relay.
- OSPF *will not* elect DRs and BDRs.
- All OSPF traffic is multicast to 224.0.0.5.
- Neighbors *do not* need to be manually specified.

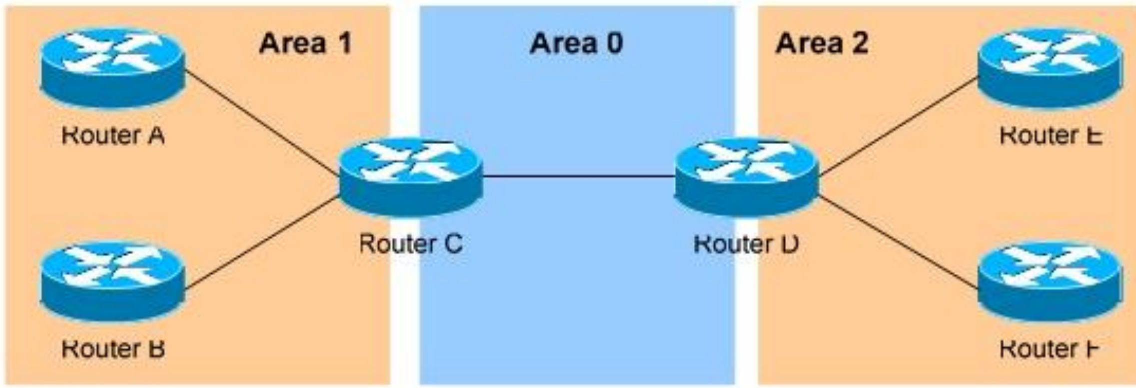
**Non-broadcast Multi-access Network (NBMA) -** indicates a topology where one interface can connect to multiple destinations; however, broadcasts cannot be sent across a NBMA network.

- An example would be Frame Relay.
- OSPF *will* elect DRs and BDRs.
- OSPF neighbors must be *manually* defined, thus All OSPF traffic is unicast instead of multicast.

**Remember:** on *non-broadcast* networks, neighbors must be **manually specified**, as multicast Hello's are not allowed.

All original material copyright © 2014 by Er. Puran S Rawat (superpuran@gmail.com), unless otherwise noted. All other material copyright © of their respective owners.
This material may be copied and used freely, but may not be altered or sold without the expressed written consent of the owner of the above copyright. Material may be found at http://www.1ccnaprojects.webs.com.

***The OSPF Hierarchy***




OSPF is a hierarchical system that separates an Autonomous System into individual **areas**. OSPF traffic can either be **intra-area** (within one area), **inter-area** (between separate areas), or **external (**from another AS).

OSPF routers build a **Topology Database** of all **links** within their area, and all routers within an area will have an *identical* topology database. Routing updates between these routers will *only* contain information about links local to their area. Limiting the topology database to include only the local area conserves bandwidth and reduces CPU loads.

**Area 0** is required for OSPF to function, and is considered the "**Backbone**" area. As a rule, all other areas must have a connection into Area 0, though this rule can be bypassed using **virtual links** (explained shortly). Area 0 is often referred to as the *transit* area to connect all other areas.

Preview from Notesale.co.uk
Page 226 of 306

OSPF routers can belong to multiple areas, and will thus contain separate Topology databases for each area. These routers are known as **Area Border Routers (ABRs)**.

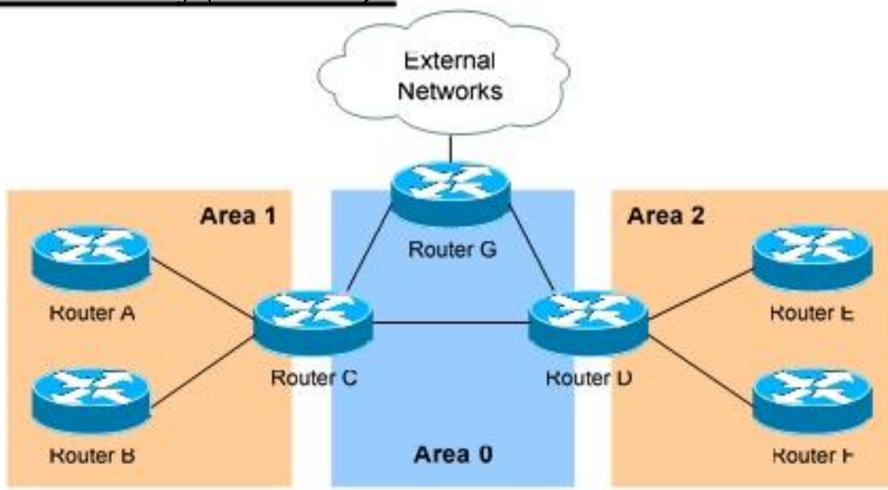
Consider the above example. Three areas exist: Area 0, Area 1, and Area 2. Area 0, again, is the backbone area for this Autonomous System. Both Area 1 and Area 2 must directly connect to Area 0.

Routers A and B belong fully to Area 1, while Routers E and F belong fully to Area 2. These are known as **Internal Routers**.

Router C belongs to both Area 0 and Area 1. Thus, it is an **ABR**. Because it has an interface in Area 0, it can also be considered a **Backbone Router**. The same can be said for Router D, as it belongs to both Area 0 and Area 2.

All original material copyright © 2014 by Er. Puran S Rawat (superpuran@gmail.com), unless otherwise noted. All other material copyright © of their respective owners.
This material may be copied and used freely, but may not be altered or sold without the expressed written consent of the owner of the above copyright. Material may be found at http://www.1ccnaprojects.webs.com.

## ***The OSPF Hierarchy (continued)***




Now consider the above example. Router G has been added, which belongs to Area 0. However, Router G also has a connection to the Internet, which is outside this Autonomous System.

This makes Router G an **Autonomous System Border Router** (**ASBR**). A router can become an ASBR in one of two ways:

- By connecting to a separate Autonomous System, such as the Internet
- By redistributing another routing protocol into the OSPF process.

ASBRs provide access to *external* networks. OSPF defines two "types" of external routes:

- **Type 2 (E2)** - Includes only the external cost to the destination network. External cost is the metric being advertised from outside the OSPF domain. This is the default type assigned to external routes.
- **Type 1 (E1)** - Includes both the external cost, and the internal cost to reach the ASBR, to determine the total metric to reach the destination network. Type 1 routes are always *preferred* over Type 2 routes to the same destination.

Thus, the four separate OSPF router types are as follows:

- **Internal Routers -** all router interfaces belong to only one Area.
- **Area Border Routers (ABRs) -** contains interfaces in at least two separate areas
- **Backbone Routers -** contain at least one interface in Area 0
- **Autonomous System Border Routers (ASBRs) -** contain a connection to a separate Autonomous System

Preview from Notesale.co.uk
Page 227 of 306

All original material copyright © 2014 by Er. Puran S Rawat (superpuran@gmail.com), unless otherwise noted. All other material copyright © of their respective owners.
This material may be copied and used freely, but may not be altered or sold without the expressed written consent of the owner of the above copyright. Material may be found at http://www.1ccnaprojects.webs.com.

## ***The OSPF Metric***

OSPF determines the best (or *shortest)* path to a destination network using a **cost** metric, which is based on the bandwidth of interfaces. The *total* cost of a route is the sum of all *outgoing* interface costs. Lowest cost is preferred.

Cisco applies default costs to specific interface types:

| ***Type*** | ***Cost*** |
|---|---|
| *Serial (56K)* | *1785* |
| *Serial (64K)* | *1562* |
| *T1 (1.544Mbps)* | *64* |
| *Token Ring (4Mbps)* | *25* |
| *Ethernet (10 Mbps)* | *10* |
| *Token Ring (16 Mbps)* | *6* |
| *Fast Ethernet* | *1* |

On Serial interfaces, OSPF will use the configured *bandwidth* (measured in Kbps) to determine the cost:

**Router(config)#** *interface s0*
**Router(config-if)#** *bandwidth 64*

Preview from Notesale.co.uk
Page 231 of 306

The default cost of an interface can be superseded:

**Router(config)#** *interface e0*
**Router(config-if)#** *ip ospf cost 5*

Changing the cost of an interface can alter which path OSPF deems the "shortest," and thus should be used with great care.

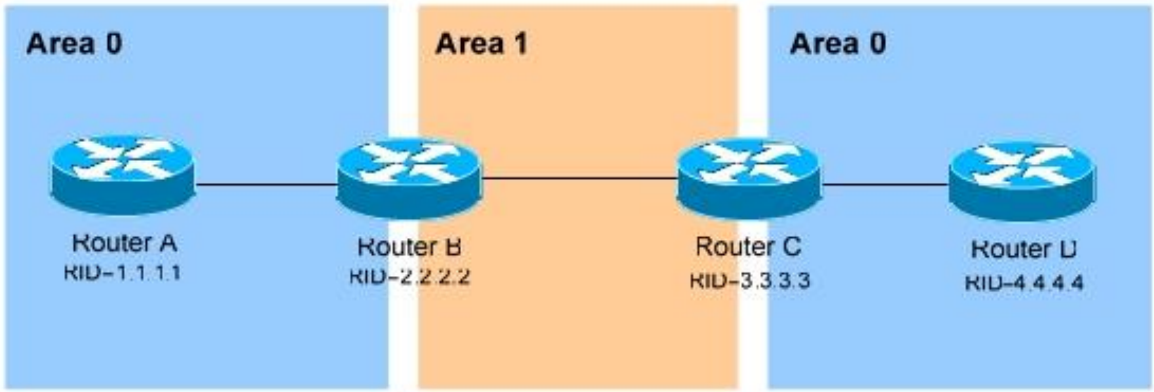
To alter how OSPF calculates its default metrics for interfaces:

**Router(config)#** *router ospf 1*
**Router(config-router)#** *ospf auto-cost reference-bandwidth 100*

The above *ospf auto-cost* command has a value of *100* configured, which is actually the default. This indicates that a 100Mbps link will have a cost of 1 (because 100/100 is 1). All other costs are based off of this. For example, the cost of 4 Mbps Token Ring is 25 because 100/4 = 25.

All original material copyright © 2014 by Er. Puran S Rawat (superpuran@gmail.com), unless otherwise noted. All other material copyright © of their respective owners.
This material may be copied and used freely, but may not be altered or sold without the expressed written consent of the owner of the above copyright. Material may be found at http://www.1ccnaprojects.webs.com.

## ***OSPF Virtual Links (continued)***




It is also possible to have two separated (or discontiguous) Area 0's. In order for OSPF to function properly, the two Area 0's must be connected using a **virtual link**.

Again, configuration occurs on the transit area's ABRs:

**RouterB(config)#** *router ospf 1*
**RouterB(config-router)#** *router-id 2.2.2.2*
**RouterB(config-router)#** *area 1 virtual-link 3.3.3.3*

**RouterC(config)#** *router ospf 1*
**RouterC(config-router)#** *router-id 3.3.3.3*
**RouterC(config-router)#** *area 1 virtual-link 2.2.2.2*

Preview from Notesale.co.uk
Page 235 of 306

**Always remember**: the area specified in the *virtual-link* command is the **transit** area. Additionally, the transit area ***cannot*** be a stub area.

As stated earlier, if authentication is enabled for Area 0, the same authentication must be configured on Virtual Links, as they are "extensions" of Area 0:

**RouterB(config)#** *router ospf 1*
**RouterB(config-router)#** *area 1 virtual-link 3.3.3.3 message-digest-key 1 md5 MYKEY*

**RouterC(config)#** *router ospf 1*
**RouterC(config-router)#** *area 1 virtual-link 2.2.2.2 message-digest-key 1 md5 MYKEY*

All original material copyright © 2014 by Er. Puran S Rawat (superpuran@gmail.com), unless otherwise noted. All other material copyright © of their respective owners.
This material may be copied and used freely, but may not be altered or sold without the expressed written consent of the owner of the above copyright. Material may be found at http://www.1ccnaprojects.webs.com.

## Troubleshooting OSPF

To view the OSPF Neighbor Table:

**Router#** *show ip ospf neighbor*

| Neighbor ID | Pri | State | Dead Time | Address | Interface |
|---|---|---|---|---|---|
| 7.7.7.7 | 1 | FULL/ - | 00:00:36 | 150.50.17.2 | Serial0 |
| 6.6.6.6 | 1 | FULL/DR | 00:00:11 | 150.50.18.1 | Ethernet0 |

The Neighbor Table provides the following information about each neighbor:

- The ***Router ID*** of the remote neighbor.
- The OSPF ***priority*** of the remote neighbor (used for DR/BDR elections).
- The current neighbor ***state***.
- The ***dead interval*** timer.
- The connecting ***IP address*** of the remote neighbor.
- The local ***interface*** connecting to the remote neighbor.

Preview from Notesale.co.uk
Page 236 of 306

To view the OSPF topology table:

**Router#** *show ip ospf database*

OSPF Router with ID (9.9.9.9) (Process ID 10)

Router Link States (Area 0)

| Link ID | ADV Router | Age | Seq# | Checksum | Link count |
|---|---|---|---|---|---|
| 7.7.7.7 | 7.7.7.7 | 329 | 0x80000007 | 0x42A0 | 2 |
| 8.8.8.8 | 8.8.8.8 | 291 | 0x80000007 | 0x9FFC | 1 |

Summary Net Link States (Area 0)

| Link ID | ADV Router | Age | Seq# | Checksum |
|---|---|---|---|---|
| 192.168.12.0 | 7.7.7.7 | 103 | 0x80000005 | 0x13E4 |
| 192.168.34.0 | 7.7.7.7 | 105 | 0x80000003 | 0x345A |

The Topology Table provides the following information:

- The actual ***link*** (or *route*).
- The ***advertising*** Router ID.
- The link-state ***age*** timer.
- The ***sequence number*** and ***checksum*** for each entry.

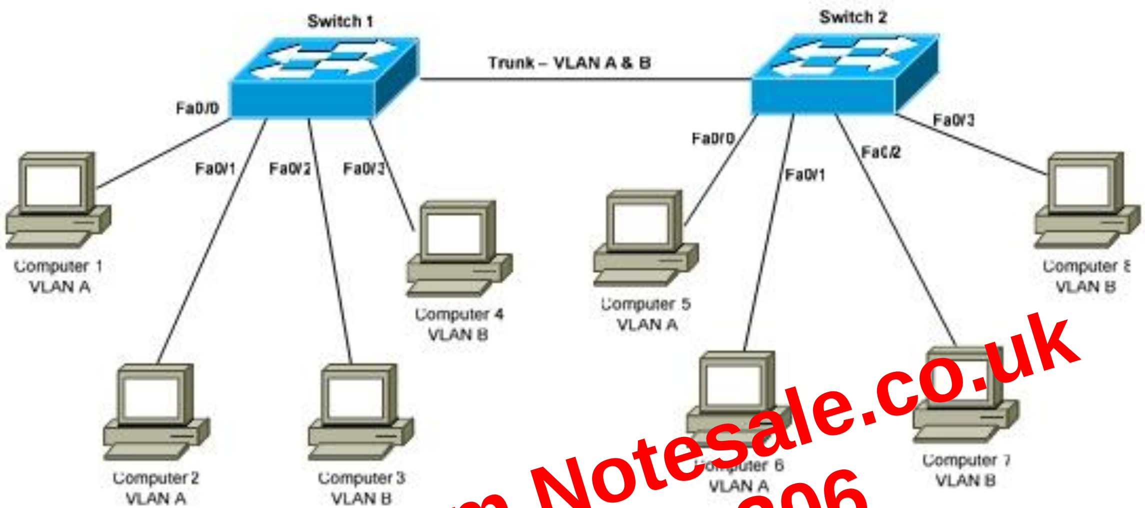
(Reference: http://www.cisco.com/en/US/products/sw/iosswrel/ps5187/products_command_reference_chapter09186a008017d02e.html)

All original material copyright © 2014 by Er. Puran S Rawat (superpuran@gmail.com), unless otherwise noted. All other material copyright © of their respective owners.
This material may be copied and used freely, but may not be altered or sold without the expressed written consent of the owner of the above copyright. Material may be found at http://www.1ccnaprojects.webs.com.

## ***VLAN Frame-Tagging***

When utilizing trunk links, switches need a mechanism to identify which VLAN a particular frame belongs to. **Frame tagging** places a VLAN ID in each frame, identifying which VLAN the frame belongs to.

Tagging occurs *only* when a frame is **sent out a trunk port**. Consider the following example:




Preview from Notesale.co.uk
Page 246 of 306

If Computer 1 sends a frame to Computer 2, no frame tagging will occur. The frame never leaves the Switch 1, stays within its own VLAN, and will simply be **switched** to Computer 2.

If Computer 1 sends a frame to Computer 3, which is in a separate VLAN, frame tagging will *still* not occur. Again, the frame never leaves the switch, but because Computer 3 is in a different VLAN, the frame must be **routed**.

If Computer 1 sends a frame to Computer 5, the frame *must* be **tagged** before it is sent out the trunk port. It is stamped with its VLAN ID (in this case, VLAN A), and when Switch 2 receives the frame, it will only forward it out ports belonging to VLAN A (fa0/0, and fa0/1). If Switch 2 has Computer 5's MAC address in its CAM table, it will only send it out the appropriate port (fa0/0).

Cisco switches support two frame-tagging protocols, **Inter-Switch Link (ISL)** and **IEEE 802.1Q.**

All original material copyright © 2014 by Er. Puran S Rawat (superpuran@gmail.com), unless otherwise noted. All other material copyright © of their respective owners.
This material may be copied and used freely, but may not be altered or sold without the expressed written consent of the owner of the above copyright. Material may be found at http://www.1ccnaprojects.webs.com.

# Section 22
# - Access Control Lists -

## ***Access Control Lists (ACLs)***

**Access control lists (ACLs)** can be used for two purposes on Cisco devices: to **filter** traffic, and to **identify** traffic.

Access lists are a set of rules, organized in a rule table. Each rule or line in an access-list provides a condition, either **permit** or **deny**:

- When using an access-list to filter traffic, a *permit* statement is used to “allow” traffic, while a *deny* statement is used to “block” traffic.
- Similarly, when using an access list to identify traffic, a *permit* statement is used to “include” traffic, while a *deny* statement states that the traffic should “not” be included. It is thus interpreted as a **true**/**false** statement.

Filtering traffic is the primary use of access lists. However, there are several instances when it is necessary to identify traffic using ACLs, including:

- Identifying interesting traffic to bring up an ISDN link or VPN tunnel
- Identifying routes to filter or allow in routing updates
- Identifying traffic for QoS purposes

Preview from Notesale.co.uk
Page 257 of 306

When filtering traffic, access lists are applied on interfaces. As a packet passes through a router, the top line of the rule list is checked first, and the router continues to go down the list until a match is made. Once a match is made, the packet is either permitted or denied.

There is an implicit ‘deny all’ at the end of all access lists. You don’t create it, and you can’t delete it. Thus, access lists that contain **only deny statements** will **prevent all traffic**.

Access lists are applied either inbound (packets received on an interface, before routing), or outbound (packets leaving an interface, *after* routing). Only one access list **per interface**, **per protocol**, **per direction** is allowed.

More specific and frequently used rules should be at the top of your access list, to optimize CPU usage. New entries to an access list are added to the bottom. You **cannot remove individual lines** from a numbered access list. You must delete and recreate the access to truly make changes. Best practice is to use a text editor to manage your access-lists.

All original material copyright © 2014 by Er. Puran S Rawat (superpuran@gmail.com),
unless otherwise noted. All other material copyright © of their respective owners.
This material may be copied and used freely, but may not be altered or sold without the expressed written consent of the owner of the above copyright. Material may be found at http://www.1ccnaprojects.webs.com.

## ***Wild Card Masks***

IP access-lists use **wildcard masks** to determine two things:

1. Which part of an address must match exactly
2. Which part of an address can match any number

This is as opposed to a **subnet mask**, which tells us what part of an address is the network (subnet), and what part of an address is the host. Wildcard masks look like inversed subnet masks.

Consider the following address and wildcard mask:

Address: 172.16.0.0
Wild Card Mask: 0.0.255.255

The above would match any address that begins “172.16.” The last two octets could be anything. How do I know this?

**Two Golden Rules of Access Lists:**

1. If a bit is set to **0** in a wild-card mask, the corresponding bit in the address must be **matched exactly**.
2. If a bit is set to **1** in a wild-card mask, the corresponding bit in the address can **match any number**. In other words, we “don't care” what number it matches.

Preview from Notesale.co.uk
Page 259 of 306

To see this more clearly, we'll convert both the address and the wildcard mask into binary:

Address: 10101100.00010000.00000000.00000000
Wild Card Mask: 00000000.00000000.11111111.11111111

Any **0** bits in the wildcard mask, indicates that the corresponding bits in the address must be matched exactly. Thus, looking at the above example, we must exactly match the following in the first two octets:

10101100.00010000 = 172.16

Any **1** bits in the wildcard mask indicates that the corresponding bits can be anything. Thus, the last two octets can be any number, and it will still match this access-list entry.

All original material copyright © 2014 by Er. Puran S Rawat (superpuran@gmail.com), unless otherwise noted. All other material copyright © of their respective owners.
This material may be copied and used freely, but may not be altered or sold without the expressed written consent of the owner of the above copyright. Material may be found at http://www.1ccnaprojects.webs.com.

## ***Wild Card Masks (continued)***

If wanted to match a **specific address** with a wildcard mask (we'll use an example of 172.16.1.1), how would we do it?

Address: 172.16.1.1
Wild Card Mask: 0.0.0.0

Written out in binary, that looks like:

Address: 10101100.00010000.00000001.00000001
Wild Card Mask: 00000000.00000000.00000000.00000000

Remember what a wildcard mask is doing. A **0** indicates it must match exactly, a **1** indicates it can match anything. The above wildcard mask has all bits set to 0, which means we must match all four octets exactly.

There are actually two ways we can match a host:

- Using a wildcard mask with all bits set to 0 - **172.16.1.1 0.0.0.0**
- Using the keyword "host" - **host 172.16.1.1**

How would we match **all addresses** with a wildcard mask?

Preview from Notesale.co.uk
Page 260 of 306

Address: 0.0.0.0
Wild Card Mask: 255.255.255.255

Written out in binary, that looks like:

Address: 00000000.00000000.00000000.00000000
Wild Card Mask: 11111111.11111111.11111111.11111111

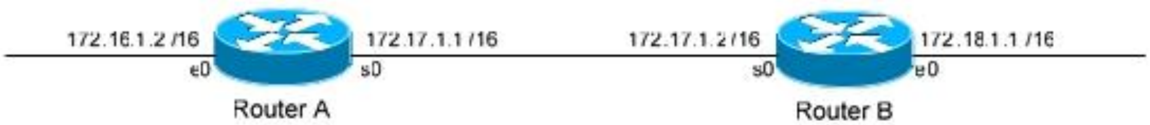
Notice that the above wildcard mask has all bits set to 1. Thus, each bit can match anything - resulting in the above address and wildcard mask matching all possible addresses.

There are actually two ways we can match all addresses:

- Using a wildcard mask with all bits set to 1 - **0.0.0.0 255.255.255.255**
- Using the keyword "any" - **any**

All original material copyright © 2014 by Er. Puran S Rawat (superpuran@gmail.com), unless otherwise noted. All other material copyright © of their respective owners.
This material may be copied and used freely, but may not be altered or sold without the expressed written consent of the owner of the above copyright. Material may be found at http://www.1ccnaprojects.webs.com.

## ***ICMP Access List***




Consider this scenario. You've been asked to block anyone from the 172.18.x.x network from "pinging" anyone on the 172.16.x.x network. You want to allow everything else, including all other ICMP packets.

The specific ICMP port that a "ping" uses is **echo**. To block specific ICMP parameters, use an extended IP access list. On Router B, we would configure:

**Router(config)#** access-list 102 deny icmp 172.18.0.0 0.0.255.255 172.16.0.0 0.0.255.255 echo
**Router(config)#** access-list 102 permit icmp 172.18.0.0 0.0.255.255 172.16.0.0 0.0.255.255
**Router(config)#** access-list 102 permit ip any any

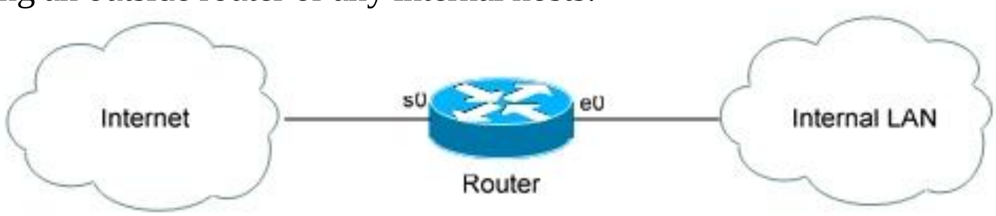
The first line blocks only ICMP echo requests (pings). The second line allows all other ICMP traffic. The third line allows all other IP traffic.

Don't forget to apply it to an interface on Router B:

**Router(config)#** *int e0*
**Router(config-if)#** *ip access-group 102 in*

Preview from Notesale.co.uk
Page 265 of 306

Untrusted networks (such as the Internet) should usually be blocked from pinging an outside router or any internal hosts:




**Router(config)#** *access-list 102 deny icmp any any*
**Router(config)#** *access-list 102 permit ip any any*

**Router(config)#** *interface s0*
**Router(config-if)#** *ip access-group 102 in*

The above access-list completed disables ICMP on the serial interface. However, this would effectively disable ICMP traffic *in both directions* on the router. Any replies to pings initiated by the Internal LAN would be blocked on the way back in.

All original material copyright © 2014 by Er. Puran S Rawat (superpuran@gmail.com), unless otherwise noted. All other material copyright © of their respective owners.
This material may be copied and used freely, but may not be altered or sold without the expressed written consent of the owner of the above copyright. Material may be found at http://www.1ccnaprojects.webs.com.

## ***Advanced Wildcard Masks (continued)***

Notice now that the *only* bits that are different between the four addresses are the last two bits. Not only that, but we use every computation of those last two bits: 00, 01, 10, 11.

Thus, since those last two bits can be anything, the last two bits of our wildcard mask are set to **1**.

The resulting access-list line:

**Router(config)#** *access-list 10 deny 172.16.1.4 0.0.0.3*

We also could have determined the appropriate address and wildcard mask by using AND/XOR logic.

To determine the address, we perform a logical **AND** operation:

1. If all bits in a column are set to **0**, the corresponding address bit is **0**
2. If all bits in a column are set to **1,** the corresponding address bit is **1**
3. If the bits in a column are a mix of **0**'s and **1**'s, the corresponding address bit is a **0.**

Observe:

| | |
|---|---|
| 172.16.1.4: | 10101100.00010000.00000001.00000100 |
| 172.16.1.5: | 10101100.00010000.00000001.00000101 |
| 172.16.1.6: | 10101100.00010000.00000001.00000110 |
| 172.16.1.7: | 10101100.00010000.00000001.00000111 |
| Result: | 10101100.00010000.00000001.00000100 |

Our resulting address is **172.16.1.4**. This gets us half of what we need.

Preview from Notesale.co.uk
Page 270 of 306

All original material copyright © 2014 by Er. Puran S Rawat (superpuran@gmail.com), unless otherwise noted. All other material copyright © of their respective owners.
This material may be copied and used freely, but may not be altered or sold without the expressed written consent of the owner of the above copyright. Material may be found at http://www.1ccnaprojects.webs.com.

***Advanced Wildcard Masks (continued)***

Two more examples. How would we deny all **odd** addresses on the 10.1.1.x/24 subnet in one access-list line?

**Router(config)#** *access-list 10 deny 10.1.1.1 0.0.0.254*

Written in binary:

| | |
|---|---|
| 10.1.1.1: | 00001010.00000001.00000001.00000001 |
| Wild Card Mask: | 00000000.00000000.00000000.11111110 |

What would the result of the above wildcard mask be?

1. The first three octets must match exactly.
2. The last bit in the fourth octet must match exactly. Because we set this bit to **1** in our address, every number this matches will be **odd**.
3. All other bits in the fourth octet can match any number.

Simple, right? How would we deny all **even** addresses on the 10.1.1.x/24 subnet in one access-list line?

**Router(config)#** *access-list 10 deny 10.1.1.0 0.0.0.254*

Written in binary:

| | |
|---|---|
| 10.1.1.0: | 00001010.00000001.00000001.00000000 |
| Wild Card Mask: | 00000000.00000000.00000000.11111110 |

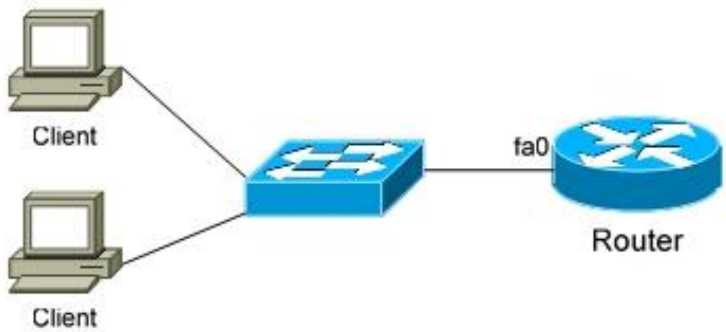
What would the result of the above wildcard mask be?

4. The first three octets must match exactly.
5. The last bit in the fourth octet must match exactly. Because we set this bit to **0** in our address, every number this matches will be **even**.
6. All other bits in the fourth octet can match any number.

Preview from Notesale.co.uk
Page 272 of 306

All original material copyright © 2014 by Er. Puran S Rawat (superpuran@gmail.com), unless otherwise noted. All other material copyright © of their respective owners. This material may be copied and used freely, but may not be altered or sold without the expressed written consent of the owner of the above copyright. Material may be found at http://www.1ccnaprojects.webs.com.

## ***Configuring a Cisco Router as a DHCP Server***




Cisco routers can be configured to function as DHCP servers. The first step is to create a DHCP **pool**:

**Router(config)#** *ip dhcp pool MYPOOL*
**Router(dhcp-config)#** *network 192.168.1.0 255.255.255.0*

The first command creates a *dhcp pool* named *MYPOOL*. The second command creates our DHCP **scope**, indicating the range of addresses to be leased. The above command indicates any address between 192.168.1.1 - 192.168.1.255 can be leased.

Specific addresses can be excluded from being leased:

**Router(config)#** *ip dhcp excluded-address 192.168.1.1*
**Router(config)#** *ip dhcp excluded-address 192.168.1.5 192.168.1.10*

Preview from Notesale.co.uk
Page 279 of 306

The first command excludes only address *192.168.1.1*. The second command excludes address *192.168.1.5* through *192.168.1.10*.

To specify DHCP options to be leased with the address:

**Router(config)#** *ip dhcp pool MYPOOL*
**Router(dhcp-config)#** *default-router 192.168.1.1*
**Router(dhcp-config)#** *dns-server 192.168.1.5*
**Router(dhcp-config)#** *domain-name MYDOMAIN*

To specify the duration of the DHCP lease:

**Router(config)#** *ip dhcp pool MYPOOL*
**Router(dhcp-config)#** *lease 1 12*

The above changes the default *lease* from 8 days to *1* day, *12* hours. To view current DHCP leases:

**Router#** *show ip dhcp binding*

All original material copyright © 2014 by Er. Puran S Rawat (superpuran@gmail.com), unless otherwise noted. All other material copyright © of their respective owners. This material may be copied and used freely, but may not be altered or sold without the expressed written consent of the owner of the above copyright. Material may be found at http://www.1ccnaprojects.webs.com.

# Part V

***WANs***

Preview from Notesale.co.uk
Page 281 of 306

All original material copyright © 2014 by Er. Puran S Rawat (superpuran@gmail.com), unless otherwise noted. All other material copyright © of their respective owners.
This material may be copied and used freely, but may not be altered or sold without the expressed written consent of the owner of the above copyright. Material may be found at http://www.1ccnaprojects.webs.com.

# Section 25
# - PPP -

## *WAN Encapsulation*

Recall that WAN technologies operate at both **Physical** and **Data-link** layers of the OSI models, and that higher-layer protocols such as IP are **encapsulated** when sent across the WAN link.

A WAN is usually terminated on a Cisco device's serial interface. Serial interfaces support a wide variety of **WAN encapsulation types**, which must be manually specified.

By default, a serial interface will utilize **HDLC** for encapsulation. Other supported encapsulation types include:

- SDLC
- PPP
- LAPB
- Frame-Relay
- X.25
- ATM

Preview from Notesale.co.uk
Page 286 of 306

Regardless of the WAN encapsulation used, it must identical on both sides of a point-to-point link.

## *HDLC Encapsulation*

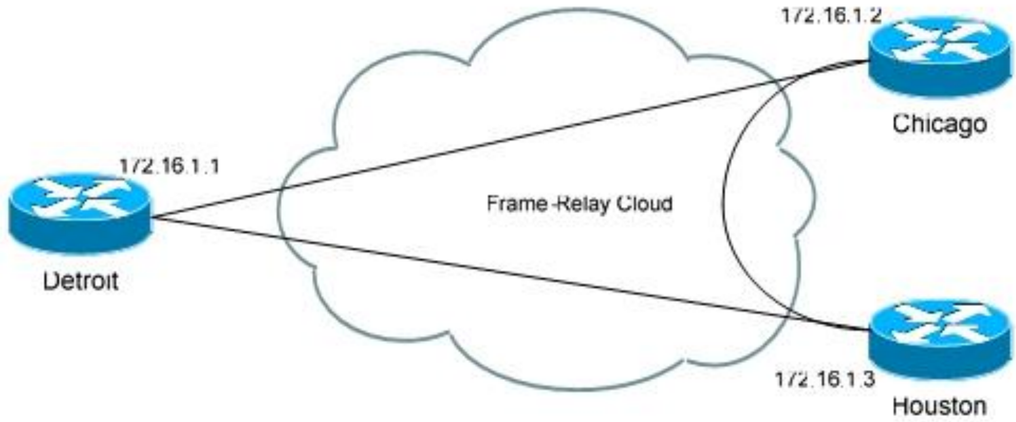
**High-Level Data-link Control (HDLC)** is a WAN encapsulation protocol used on dedicated point-to-point serial lines.

Though HDLC is technically an ISO standard protocol, Cisco's implementation of HDLC is proprietary, and will not work with other routers.

HDLC is also Cisco's **default encapsulation** type for serial point-to-point links. HDLC provides *no* authentication mechanism.

All original material copyright © 2014 by Er. Puran S Rawat (superpuran@gmail.com), unless otherwise noted. All other material copyright © of their respective owners.
This material may be copied and used freely, but may not be altered or sold without the expressed written consent of the owner of the above copyright. Material may be found at http://www.1ccnaprojects.webs.com.

## Frame-Relay Full Mesh Configuration Example




Consider the above example, a full mesh between three locations. All routers can still belong to the same IP subnet; however, DLCI's must now be *mapped* to IP addresses, as multiple PVCs are necessary on each interface.

This can be dynamically configured via **Inverse-Arp**, which is enabled by default (as stated earlier). Otherwise, the DLCI-to-IP mapping can be performed manually. Looking at the Detroit and Chicago router's configuration:

Preview from Notesale.co.uk
Page 294 of 306

Detroit Router:

```
Router(config)#  int s0/0
Router(config-if)#  ip address 172.16.1.1 255.255.0.0
Router(config-if)#  encapsulation frame-relay ietf
Router(config-if)#  no frame-relay inverse-arp
Router(config-if)#  frame-relay lmi-type ansi
Router(config-if)#  frame-relay map ip 172.16.1.2 102 broadcast
Router(config-if)#  frame-relay map ip 172.16.1.3 103 broadcast
Router(config-if)#  no shut
```

Chicago Router:

```
Router(config)#  int s0/0
Router(config-if)#  ip address 172.16.1.2 255.255.0.0
Router(config-if)#  encapsulation frame-relay ietf
Router(config-if)#  no frame-relay inverse-arp
Router(config-if)#  frame-relay lmi-type ansi
Router(config-if)#  frame-relay map ip 172.16.1.1 201 broadcast
Router(config-if)#  frame-relay map ip 172.16.1.3 203 broadcast
Router(config-if)#  no shut
```

Inverse-ARP was disabled using the *no frame-relay inverse-arp* command.

The *frame-relay map* command maps the remote router's IP address to a DLCI. On the Detroit router, a *map* was created to Chicago's IP (*172.16.1.2*), and that PVC was assigned a DLCI of *102*. The *broadcast* option allows broadcasts and multicasts to be forwarded to that address, so that routing protocols such as OSPF can form neighbor relationships.

All original material copyright © 2014 by Er. Puran S Rawat (superpuran@gmail.com), unless otherwise noted. All other material copyright © of their respective owners.
This material may be copied and used freely, but may not be altered or sold without the expressed written consent of the owner of the above copyright. Material may be found at http://www.1ccnaprojects.webs.com.

## ***Configuring Frame-Relay (FRTS)***

To configure FRTS, a **map-class** must be created:

**Router(config)#** *map-class frame-relay MYCLASS*
**Router(config-map-class)#** *frame-relay cir 64000*
**Router(config-map-class)#** *frame-relay bc 8000*
**Router(config-map-class)#** *frame-relay be 0*
**Router(config-map-class)#** *frame-relay mincir 32000*
**Router(config-map-class)#** *frame-relay adaptive-shaping becn*

A *map-class* was created for *frame-relay* called *MYCLASS*. The first three commands configure the *CIR, Bc,* and *Be* respectively.

The final commands must be used in conjunction with each other. The *adaptive-shaping* feature has been specified, indicating that the router will throttle back to the *mincir* if a *becn* is received. The router does not throttle down to the *mincir* immediately, but rather will lower the rate by 25% until either the congestion stops, or the *mincir* is reached.

A map-class applied to an interface affects *all* PVCs on that interface. Additionally, map classes can be applied to a specific PVC, providing more granular control of FRTS.

Preview from Notesale.co.uk
Page 298 of 306

To apply a map class to an interface:

**Router(config)#** *interface s0/0*
**Router(config-if)#** *encapsulation frame-relay*
**Router(config-if)#** *frame-relay traffic-shaping*
**Router(config-if)#** *frame-relay class MYCLASS*

To apply a map class to a specific PVC:

**Router(config)#** *interface s0/0*
**Router(config-if)#** *encapsulation frame-relay*
**Router(config-if)#** *frame-relay traffic-shaping*
**Router(config-if)#** *frame-relay interface-dlci 101 class MYCLASS*

Do not forget the *frame-relay traffic-shaping* command. Once this command is configured, all PVCs are configured with the default CIR of 56,000 bps.

All original material copyright © 2014 by Er. Puran S Rawat (superpuran@gmail.com), unless otherwise noted. All other material copyright © of their respective owners.
This material may be copied and used freely, but may not be altered or sold without the expressed written consent of the owner of the above copyright. Material may be found at http://www.1ccnaprojects.webs.com.